数据科学基础
(With Python)

关联规则

频繁模式

- 频繁模式: 在数据集中频繁出现的模式
- 动机:找出数据中的规律
  - 哪些产品经常被一起购买?
    - Beer and diapers?!
  - 购买了PC之后下一个会买什么?
- 应用
  - 购物篮分析
  - 购物网站推荐
  - Web日志分析, ...

基本概念

- 项目(物品)集合$I = \{i_1, \ldots, i_m\}$
  - I的子集称为项集
- 交易集$D = \{T_i \mid T_i \subseteq I\}$
  - 每个交易有唯一TID
- 目的:找出在D中频繁出现的项集X
  - 支持度supp(X):包含X的交易数(频数)
    - 也可以用占总交易数的比例(即频率)
  - 预先指定最小支持度min_sup,则支持度大于min_sup的项集称为频繁的

基本概念

- 关联规则:$X \Rightarrow Y$,其中X和Y是不相交项集
  - 支持度$supp(X \Rightarrow Y) = supp(X \cup Y)$
    - 也可说是$X \cup Y$在D中出现的概率
  - 置信度$conf(X \Rightarrow Y) = supp(X \cup Y)/supp(X)$
    - 也可说是条件概率$P(Y|X)$
- 强关联规则:支持度超过预定的min_sup且置信度超过预定的min_conf
  - 即$X \cup Y$是频繁的
  - 因此:为找强关联规则,可先找频繁项集

例:基本概念

- 令sup_min = 50%，min_conf = 50%
- 频繁项集{A,D}:支持度=3/5≥50%)
- (强)关联规则:
  - A⇒D (60%, 100%)
  - D⇒A (60%, 75%)

Transaction-id	Itemset
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

如何找出关联规则

- 第一步:找出所有频繁项集
- 第二步:从频繁项集生成强关联规则
  - 这一步很容易
  - 例:如果X是频繁的且至少含有2个项,则将X分成任意两个不相交子集A和B,即可构造关联规则$A \Rightarrow B$和$B \Rightarrow A$
    - 这两条关联规则的支持度即X的支持度,显然≥min_sup,只需要验证置信度是否≥min_conf

如何找频繁项集

- 似乎也很容易?
- 蛮力算法:
  - 考虑I的所有可能子集:

    $\{i_1\},\{i_2\},...,\{i_m\},\{i_1,i_2\},...,\{i_1,i_m\},...,\{i_1,...,i_m\}$
  - 扫描D,对每个交易检查是否包含上述子集,是则给该子集计数+1
  - 扫描完毕,则所有子集的支持度已知,超过min_sup的项集即为频繁项集
- 问题是I的子集有2^m个,而m可能成百上千

频繁项集的一个性质

- 频繁项集的任何非空子集必是频繁的
  - 若{A,B,C}是频繁的, 则{A}, {B}, {C}, {A,B}, {A,C}, {B,C}也是频繁的.
    - 因为任何包含{A,B,C}的交易也包含它的任意子集,即:子集的支持度≥超集的支持度
- 启示:
  - 首先找频繁1-项集(全体记为L_1),
  - 然后利用L_1构造频繁2-项集(全体记为L_2)
  - 利用L_2构造频繁3-项集(全体记为L_3) ...

从L_k构造L_{k+1}

- 连接:L_k join L_k得到k+1项集的集合C_{k+1}
  - 约定:所有项按字母顺序排序
  - 设X_1和X_2是L_k中的两个频繁k项集,且前k-1个项是相同的,仅第k个项不同.则X_1和X_2连接得到一个k+1项集:

    $\{X_1[1], X_1[2],..., X_1[k-1], X_1[k], X_2[k]\}$
- 检测C_{k+1}中频繁k+1项集,构成L_{k+1}
  - 扫描一遍D即可
  - 优化:某元素若有k子集不在L_k中则可删除

Apriori算法

L_1 = {频繁1-项集};

for (k = 1; L_k !=∅; k++) do

  C_{k+1} = L_k自连接生成的候选项集

  删除C_{k+1}中有非频繁k-项集的项集

  for t ∈ D do

    将被t 包含的C_{k+1}中项集的计数加1

  L_{k+1} = C_{k+1}中至少具有min_sup的项集

return $\bigcup_k L_k$;

例:Apriori算法

min_sup = 2

Tid	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

扫描D

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

连接

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

扫描D

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

连接

C_3

Itemset
{B, C, E}

扫描D

L_3

Itemset	sup
{B, C, E}	2

例:自连接与剪枝

- 候选生成中的优化
  - **L_3 = {abc, abd, acd, ace, bcd}**
  - 自连接:
    - **abc和abd连接:abcd**
    - **acd和ace连接:acde**
    - **为什么不abd和bcd连接?**
  - 剪枝
    - **因为acde的子集ade不在L_3中,故删除acde**
  - **最终C_4={abcd}**

例:生成关联规则

- **{A,C}是频繁2-项集,可生成**
  - **A⇒C**
    - **置信度:supp({A,C})/supp({A})=2/2=100%**
  - **C⇒A**
    - **置信度:supp({A,C})/supp({C})=2/3=67%**
- **{B,C,E}是频繁3-项集,可生成**
  - **BC⇒E**
  - **BE⇒C**
  - **B⇒CE,...**

Apriori评价

- 挑战
  - 多次扫描D
  - 大量的候选项集
  - 对候选项集支持度的计数负担重
- 改进
  - 减少扫描D的遍数
  - 减少候选的个数
  - 使候选支持度计数便利

变种:多层关联规则

- 项层次
  - 例如:电脑-台式和笔记本-各品牌;软件-办公和杀毒-...
- 交易:低抽象级
  - 强关联规则较罕见,因为较低层项具有较低的支持度
- 解决方法:支持不同抽象级上的关联规则
  - 联想笔记本⇒360杀毒软件:可能支持度不够
  - 笔记本⇒杀毒软件:可能支持度够高

变种:多维关联规则

- 前述关联规则可认为是一维(单一谓词)的
  - 例如: **buy(A)⇒buy(C)**
- 对多维数据集可挖掘多维关联规则
  - 例如: **age("20..30") ∧ major("CS") ⇒buy("laptop")**

变种:定量关联规则

- 多维关联规则中,数值型特征在挖掘过程中动态地离散化,以满足某种条件(如嘉华置信度)
- 例如: 关联规则聚类

  age(34) ∧ income(31K..40K) ⇒buy("laptop")

  age(35) ∧ income(31K..40K) ⇒buy("laptop")

  age(34) ∧ income(41K..50K) ⇒buy("laptop")

  age(35) ∧ income(41K..50K) ⇒buy("laptop")

  可聚类为

  age(34..35) ∧ income(31K..50K) ⇒buy("laptop")

关联与相关性

- 关联规则基于支持度-置信度框架
  - 不一定有相关性
    - 例如:啤酒和尿布
  - **min_sup**阈值尽管能排除无意义的关联规则,但即使是强关联规则也可能没有意义
- 解决方法:将支持度-置信度框架与相关性分析等统计方法相结合

例:关联vs相关性

- 在10000个学生中
  - 6000人晨练
  - 7500人早餐吃肉包子
  - 4000既晨练又吃肉包子
- 则:晨练⇒吃肉包子[40%, 66.7%]
  - 看似有意义,其实是误导. 因为吃肉包子的学生本身就有75%, 高于66.7%.说明晨练与吃肉包子具有负相关,即晨练⇒不吃肉包子[20%, 33.3%]更有意义, 尽管具有较低支持度和置信度

End